

Estructuras de Datos y Algoritmos

Grado en Ingeniería Informática





Práctica 3:

EXCEPCIONES

OBJETIVO

- Aprender el uso de las excepciones en C++.
- Implementación de precondiciones mediante el uso de excepciones.
- Diseño de funciones recursivas.

BIBLIOGRAFÍA

• Deitel, "C++ Como programar", 6ª edición, Ed. Prentice Hall. (Capítulo 16)

MATERIAL A ENTREGAR

- pila.h, pila.cpp, error.h, prueba.cpp, rec.cpp
- makefile
- Ejercicios apartado 1 (previo).

DESARROLLO

1. Ejercicios previos

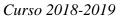
Estudiar el capítulo 16 del Deitel 6ª ed., apartados 1,2,3,4 y 13.

Escribir una clase *Error* que implemente una excepción. Debe heredar de *runtime_error* y mostrar el mensaje "Se ha producido un error". Se deben incluir los ficheros de cabecera necesarios.

Departament d'Informàtica

Estructuras de Datos y Algoritmos

Grado en Ingeniería Informática





2. Clase Pila

Implementar la clase **Pila** mediante un vector de enteros tipo C (no STL). Esta clase **CONTENDRÁ** ÚNICAMENTE el **constructor** y **destructor** de la clase, **constructor de copia**, operador de **asignación** y los métodos siguientes:

- **void apilar(int)** Apila un entero.
- **void desapilar**() Desapila la cima.
- int cima() Devuelve el elemento de la cima. No desapila.
- bool vacia() Indica si la pila está vacía.
- unsigned getAccesos() Devuelve el número de accesos realizados a la pila.
- void resetAccesos() Pone a cero el número de accesos.
- void mostrar() Muestra la pila.

El tamaño máximo de la pila se indicará en el constructor. Esto implica que la clase deberá contener un puntero a entero y que el vector se deberá crear en el constructor mediante un *new*. Las precondiciones de los métodos de la pila, en vez de implementarse mediante *assert()*, se implementarán mediante *excepciones*. Habrán **2 excepciones distintas**, excepción por pila vacía y por pila llena. Las 2 se deberán declarar en el fichero *error.h*.

La clase Pila también llevará la cuenta de cuantas veces se accede a sus elementos, es decir, cuantas veces se ejecutan los métodos *apilar desapilar* o *cima*.

Implementar una función main (*prueba.cpp*) que compruebe el funcionamiento de la clase Pila. Para ello deberá realizar lo siguiente:

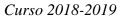
- 1. Crear una Pila con máximo de 10 elementos.
- 2. Desapilar 1 elemento.
- 3. Introducir los naturales del 1 al 11.
- 4. Mostrar la pila.
- 5. Mostrar el número de accesos realizados.

En los puntos adecuados se deberán capturar las excepciones y mostrarlas por pantalla. Después de mostrar la excepción el programa deberá continuar con el siguiente punto.

Departament d'Informàtica

Estructuras de Datos y Algoritmos

Grado en Ingeniería Informática





3. Funciones recursivas

Implementar las siguientes funciones recursivas sobre pilas. Es conveniente escribirlas primero en pseudocodigo.

- 1. int altura(Pila) Devuelve el número de elementos en la pila.
- 2. **int altura2(Pila&)** Devuelve el número de elementos en la pila.
- 3. int suma(Pila&) Devuelve la suma de todos los elementos de la pila.
- 4. **int sumaPares(Pila&)** Suma los elementos en posiciones pares de la pila, teniendo en cuenta que la cima es el primer elemento (posición impar).
- 5. **Pila borrar(Pila, Elem, Nat n)** Esta función devuelve una pila donde se han borrado las **n** primeras apariciones del elemento en la pila.

Para evitar que las funciones a las que se les pasa una pila por referencia modifiquen dicha pila, se deberá realizar una copia de la pila **antes** de la llamada. Sería conveniente que esta copia se realizara dentro de una función auxiliar.

Todas las funciones deberán implementarse en el fichero *rec.cpp*, que también contendrá una función *main* para comprobar el funcionamiento de cada una de las funciones.

El fichero makefile de la práctica deberá crear los 2 ejecutables únicamente llamando al make.

¿Qué función es más eficiente, altura o altura2 ? ¿Por qué razón? ¿En cuál se realizan más accesos?